A Dynamic Design Strategy for Visual and Haptic Development

Arthurine Breckenridge
Interaction Laboratory[1]
Sandia National Laboratories
arbreck@sandia.gov

Ben Hamlet
Sandia National Laboratories
brhamle@sandia.gov

Derek Mehlhorn
University of Washington
dtmehlh@sandia.gov

Kevin Oishi
Carnegie Mellon University
ktoishi@sandia.gov

Abstract

The evolution of modern computer programming languages comes with the need for the strategies with which we implement them to change as well. Fully dynamic and reusable visual and haptic simulations are now possible given the modular nature of current programming languages. The new standards for simulations are dynamic applications that can load and utilize code modules without shutting down, and that will almost never require a complete rebuild for new applications. Two major issues we addressed and implemented in pursuit of this standard are:

I. Managing the resources of one machine

II. Dynamic Human Computer Interface (HCI)

This paper discusses the limitations of current development strategies, and presents the work done at Sandia National Laboratories to develop an application that will conform to the new dynamic standards.

1. Limitations of Current Simulation Strategies

Current Virtual Reality (VR) simulation strategies have two major limitations. First, they do not exceed the resources of one machine. The current strategy for VR simulations of all kinds is to create a specifically tailored application, from the ground up, to accomplish a specific goal. Not only are these applications designed to simulate only one situation or event, but, once completed, they cannot be updated or modified without being completely recompiled. There are two main drawbacks to this development strategy. The applications, or parts of them, are difficult to reuse in future work, and the future simulations must be almost completely re-written. A number of attempts have been made to reduce the need for complete program rewrites. These come in the form of Application Programming Interfaces (API's).

Second, current VR strategies do not support dynamic Human Computer Interaction (HCI). HCI has traditionally been limited to 2-dimensional devices like mice or keyboards. When working with a 3-Dimensional simulation or programming environment, working with a 2-dimensional input device is counter intuitive and limiting to the user. Development in the field of computer haptics has improved HCI in a number of ways, including a much more intuitive virtual environment. Computer haptics has great potential in the VR field, not only for simulation purposes but for simulation development [1] as well. Such devices as SensAble Technologies' Desktop PHANToM[2] [2] provide a user with 6

[1] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000.

[2] PHANToM is a registered trademark of SensAble Technologies, Inc.

degrees of freedom of motion (x, y, z, yaw, pitch, and roll) as well as 3 degrees of force feedback (x, y, z). Not only do haptics devices, such as the PHANToM, allow users to interact with a computer in 3-dimensions, but they also allow the computer to interact with the user physically, in addition to the standard visual feedback. These devices, being developed into new APIs, have proven to be intuitive and efficient in the development and interaction with virtual environments [1,3]. The improvement of HCI and the emergence of API's have increased programming productivity and reusability, proving a more efficient and desirable design strategy is becoming possible.

2. A Dynamic Standard

In the past, simulation development has been limited by the linear nature of early programming languages, such as Fortran and Cobalt. Newer languages, such as C++ and Java, are now object oriented, encouraging modular code development, and support for greater code reusability. The modular nature of current programming languages lends itself well to a new strategy for VR and application design.

One of the limitations, needing to be addressed, of current VR strategies is the lack of reusability. Extremely powerful and impressive applications and simulations have been developed as single use applications. Therefore, these program's contributions to their fields are static. They cannot be easily modified to perform different tasks, nor can they be easily disassembled and applied directly to another application. Current computer hardware and software technology make possible a number of improvements to VR development strategies, including the ability to manage the resources of a machine through dynamic loading.

Simulations can now be designed in a modular nature that allows for dynamic scalability and reusability. The new standard for VR development should be multiple use applications that can dynamically load new features without recompilation. Current API's should be designed in this manner to increase productivity and reusability. The primary benefit of a new dynamic standard is the reusability and expandability that will allow for more robust and extensive simulations than ever before. Dynamically loadable modules can also enable the simulations to be run on a much larger range of machines, since users can dynamically customize the resolution, or complexity, of the application before or during execution. People using the simulation will only load the modules they need or can support with their machine. This strategy will also enable a large number of people to develop VR simulations without being programming or design experts.

The second improvement to current VR development strategies that we worked with lies in HCI. One limitation of current HCI is that, for the most part, it is limited to a single user with a static user interface. Currently, the means by which users interact with their virtual environments do not change with data needs. Therefore, in order to maximize the breadth of an application, all conceivable features are loaded upon execution, a method that will quickly exceed system resources. Our implementation of a dynamic HCI includes the ability to dynamically change the way that one can interact with one's simulation environment by using a series of dynamic menus that can be developed during runtime. The ability for multiple users to work collaboratively over a network in a Virtual Collaborative Environment (VCE) would also be extremely valuable, and has limitless potential. Productivity and quality of products will be increased as designers and customers collaboratively design in a VCE, even if they are hundreds of miles apart.

3. An Implementation Example

3.1 Overview

An initial attempt to implement the new standards of VR development, as stated above, has commenced at Sandia National Laboratories' Interaction Lab. The basic methods and strategies used are applicable to more general cases of VR development. It should be noted that the machines used in development were little more than commercial-off-the-shelf PC systems, readily available to most individuals interested in the new wave of VR development. Novint Technologies' e-Touch[3] was used as a basis for haptics development with Sensable Technologies' Desktop PHANToM device. Novint is an internet spin-off company, from the Sandia National Laboratories' Interaction Lab. e-Touch is an open module effort to provide a 3D application programming interface as well as a graphic/haptic user interface (GHUI) to the internet community.

[3] e-Touch is a registered trademark of Novint Technologies, Inc. http://www.etouch3d.org

The project is tailored towards an architectural design application. However, since the project is meant to implement the new standards for haptics and visual simulations, the application area goes far beyond a simple CAD program. At the core of the project is the need to manage the resources of a machine, to dynamically add features, ranging from rendered objects to new modules of code, without having to exit the application. Other goals include improving our HCI by supporting multiple users, working collaboratively across a network, using dynamically changing tools and interfaces.

exist within the foundation of the application. New objects, tools, and commands can then be derived from these classes, providing expandability and compatibility. This structure also enables novice users to implement basic features and provides advanced users with the ability to overload the base implementation and create unique and complex additions to the simulation. Objects and tools are also designed in a way that they depend only on the lowest level of our API, Novint's e-Touch. Command modules are used to interface them with specific applications. Therefore, the same objects and tools can be reused in an infinite array of e-Touch

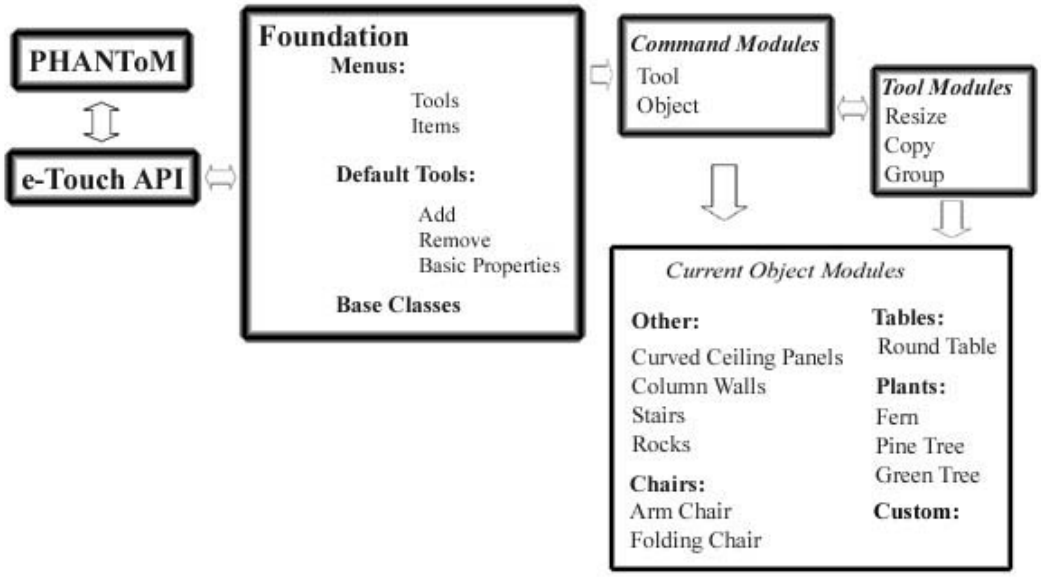

Chart 1. **Shows the design structure of the application.**

3.2 Basic Structure

The application structure breaks down into three major categories: objects, tools, and commands. Objects are graphic and/or haptic items that can be dynamically added, removed, and manipulated within the simulation. Tools enable a user to interact with one's environment and with the objects therein. The tools, except for a select few default tools, and all objects are designed to be independent code modules that can be dynamically loaded upon request. Commands are code modules that are used to interface objects and tools with the foundation of the application.

To support expandability and dynamic loading of different features, a series of base or template classes

based applications by simply swapping out their command modules.

Chart 1 shows the basic structure of our application. The foundation classes consist of, essentially, the groundwork for objects, menus, and tools. An object is defined as anything rendered with either graphics, haptics, or both. For our purposes in the architectural application, objects are created to use both styles of rendering. Tools are usually represented by a 3D cursor, which is directly controlled by the PHANToM for input. Tools are used to alter the state of the application by adding, copying, deleting, and in other ways manipulating an object. Tools generally manipulate or change some property of an object. Menus are completely three dimensional, and contain 3D objects, such as buttons,

sliders, and knobs. An explanation of the menuing system can be found in section *3.4 Menus*.

At this point, it should be noted that this project's system of combining graphical and haptic representations of objects did not require the redevelopment of 3D modeling applications. The aim is not to replace existing, proven, and refined applications, but rather to address a new method of efficient development that can use and improve upon existing programs. The graphical nature of objects is still designed using more specific, existing, applications. The project currently uses imported 3D Studio Max[4] files to create the graphics for our haptic objects, and contains a framework for converting other types of 3D object files into data that can be used by the application. Using these pre-designed graphics, a haptic representation is either specifically defined or generally applied based on a series of parameters, yielding a completed haptic object.

An important concern with any system that combines haptics and graphics data is the need for an on disk storage system that also ties these two representations together. This can be accomplished through the use of a common file format that stores both object depictions in a single file. Such a format would allow even the most novice users to create simple objects, while more advanced and knowledgeable users would still be able to create the complex, individualized, and ornate objects they require.

3.3 Tools

Our application provides a dynamic and 3D GHUI. A series of default tools allow users to interact with their environment instantly in several basic ways, while a series of 3D menus dynamically reconfigure themselves to incorporate new tools during runtime. *Image 1* shows an object being resized using one of the applications default tools. *Image 2* shows the default property manipulation tool. Without any additions to the applications, default tools enable users to add, delete, resize, and manipulate object properties, while still possessing the ability for custom tools to be loaded in at anytime for customized interaction. This system increases productivity, decreases lost time, and allows for a more robust and overall user-friendly application.

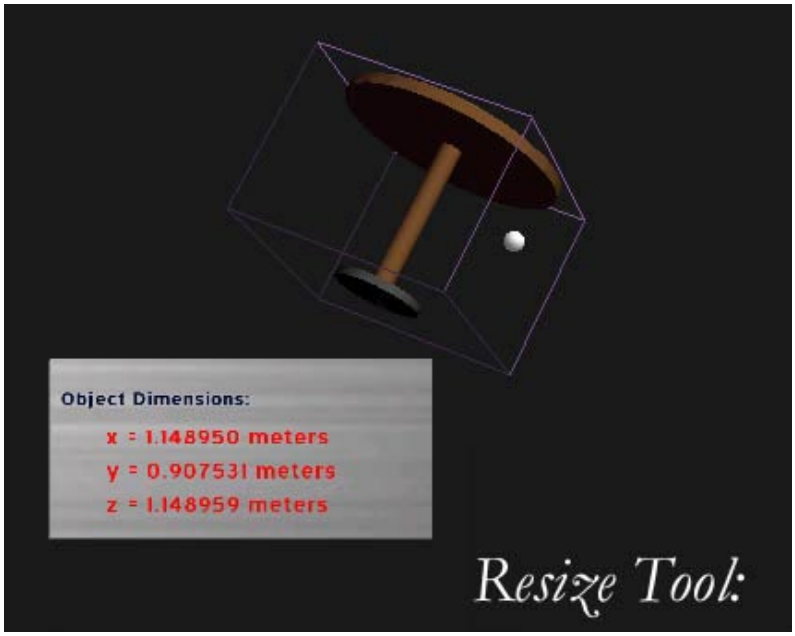

Image 1. **Shows an object being dynamically resized using the default resize tool.**

The range of properties that can be manipulated is also dynamic, allowing users to specify unique properties that they wish to change for only select types of objects. Currently, the application enables manipulation of both the size and color scheme of an object while the program is running.

[4] 3D Studio Max is a registered trademark of Autodesk, Inc.

In a system that allows for real-time extendibility, it is necessary for applications to make use of dynamic tools, and dynamic menus to interface with these tools. It is imperative for various methods of interacting with the virtual environment to be added as required. This interaction should also be able to take place in a way never imagined by the initial developers of the system, and should not be limited to any basic, underlying, and restricting subsystem.

A method for completing such a task has been implemented for this project. Several tools have been added without restructuring or altering the foundation of the application. An example of this is a grouping tool, which allows users to group together any number of objects. Then, any change made to an object in a group is applied to all other objects in that same group. Such tools are derived off a base tool class, which contains only those properties common to all tools, and a means to interface with e-Touch. They are independent of the foundation application, and are only specific to the very general object type that is used throughout the application. This proves the ability of the application, but more importantly, the new standard of development, to allow for the kind of extendibility warranted and required by such an ambitious project.

3.4 Menus

A menu system is another aspect of this project. An important part of this feature is the three dimensional nature of the menus, whose basic implementation is in the e-Touch API. Although visually similar to older windowing systems, giving them a sense of familiarity, these menus are quickly realized to be much more functional than their 2D counterparts. Menu objects, such as buttons and sliders, are also in 3D. Using the PHANToM haptic device, sliders are moved, buttons are clicked, or knobs are turned to adjust aspects of an object. This is a very powerful feature when coupled with a device such as the PHANToM, as buttons and other objects have a physical response when they are activated. Users can feel 3D buttons being pushed, similar to their real world equivalents, such as the buttons on a telephone. A particularly useful example is the 3D slider used for changing the color of an object. A wire frame cube is used to represent all available colors. The slider can be positioned by the user at any point in the cube, and the object's color is determined by the slider's location, see *Image 2.*

Image 2. **Shows an object's color property being manipulated using the default properties tool and it's corresponding dynamic menu.**

As our particular application strives to exceed the resources of a single machine, which basically means it needs to be expandable at runtime, a dynamic menuing system was built on top of a set of default, foundation menus. An application area where users are able to easily add a variety of objects to the simulation is needed. This is accomplished by way of a menu system that recognizes the addition of new objects and enables their use. Specifically, a particular folder in the application directory has been set aside for storage of objects the program may use. When a user decides to add an object, this directory is scanned for subfolders, which are presented as buttons on an object adding toolbar. When a button on this toolbar is pressed, a corresponding directory is scanned, and, dynamically, a new object selection menu is built. Users then choose which object they wish to add by pressing any of the buttons on the menu. The menu is destroyed when closed, allowing for new source files for objects to be added to the program directory at runtime. These objects will then be represented in the menus the next time an object is to be added. The menuing system is depicted in *Image 3*. Currently, the project has a dynamic menu that recognizes the addition of new objects using a general haptic object file format and allows the user to use new objects without halting program execution. The goal of dynamic loading without shutting down is aimed toward being able to access features from the internet dynamically.

3.5 Lessons Learned

Developing an application that conforms to the new standard for visual and haptic development has not been a small task. Although modular programming languages allow programmers to write code that maintains a structure that can support dynamic loading, there is not currently a good program that allows for dynamic loading of code modules. The most promising is a program called Bamboo [6]. Other options include placing code modules within dll's (dynamic linking libraries). This enables both dynamic linking, adding code at the start of execution, and dynamic loading, adding code during execution.

In order to improve current HCI, we were working with SensAble Technologies' Desktop PHANToM. During our work we discovered several limitations of the current haptic technology. For example, a limited range of motion potentially requires scaling of the device motion. Other recognized haptics needs

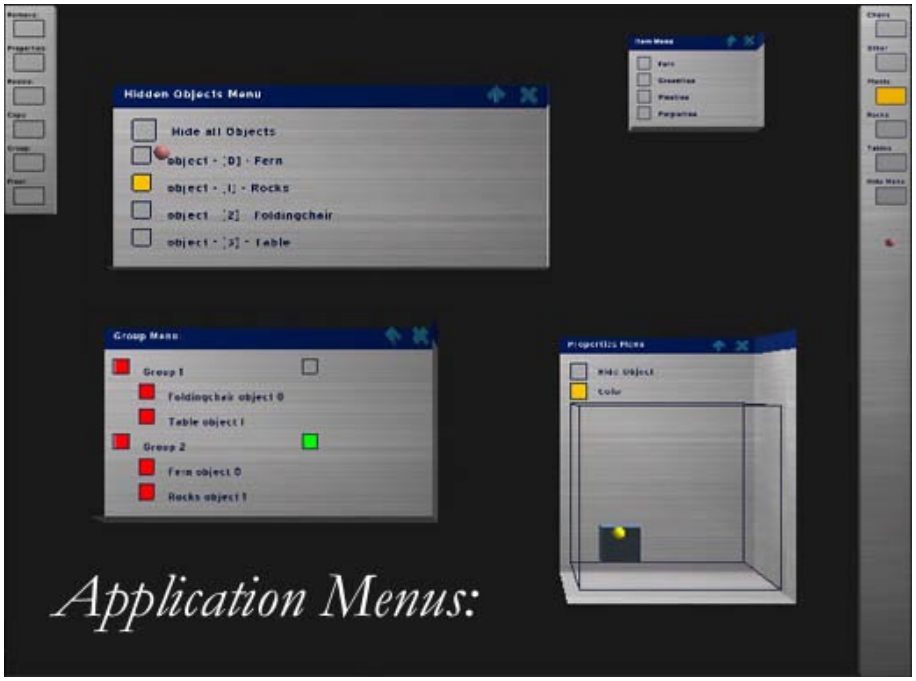

Image 3. **Shows a number of dynamic menus that make up our unique HCI.**

involve a standard format for object property definition. Since graphics and haptics are produced by different output devices, a monitor versus a PHANToM, they have always been treated as two separate entities. Creating a standard format for haptic object information is a necessary step in the development and continuation of the computer haptics field. A behavior constraint library that can be referenced and used to define behaviors and characteristics of specific objects is one promising approach to a common file format. Investigation and communication with the haptics community at large is required for this format to be as inclusive and useful as possible.

Issues to consider regarding supporting multiple users to improve and develop HCI include, but are not limited to, network latency and the effects it has on force feedback[5,6], and platform independence. Haptics is difficult to successfully perform over a network due to the high refresh rates and the subsequent and inherent sensitivity of such simulations. Also, because of the complicated nature of haptics, threads are inexorably an integral part of any simulation. Due to the often platform specific nature of threads, this is a complex and inevitable problem encountered with networking individuals using various system configurations. In our goal for reusable and modular code, platform independence is an important obstacle to overcome.

4. Conclusions

Given the evolution of programming languages and techniques, it is possible for more efficient and productive VR applications to be designed. It is now possible to exceed the resources of one's machine and to improve and develop current HCI methods. Specifically, multiple use, multiple user simulations that are composed of dynamically loadable modules, have become a reality and present a bright future for VR development and expansion.

An architectural design application based on and implementing the concepts and ideas presented in this paper is currently under development at Sandia National Laboratories. The foundation of the application has been completed utilizing Novint Technologies' haptics API e-Touch. We have been successful in importing various unique objects from 3D Studio Max files into our simulation, and are able to manipulate their location, orientation, and physical properties dynamically with a number of tools that have been written into the application foundation. The modular, and eventually dynamically loadable, structure of our application has also been verified using several independent tool modules that successfully interact with objects inside the simulation.

The work at Sandia National Laboratories already shows great potential for the full implementation of the ambitious VR standards set forth in this paper. Dynamically expandable, multiple user simulations with a high degree of reusability are on the threshold of reality, and are the future of VR design and development.

5. References

[1] Anderson, Tom, "FLIGHT: A 3D Human-Computer Interface and Application Development Environment", Proceedings of the Second PHANTOM Users Group Workshop, Cambridge, MA, 1997

[2] SensAble Devices Inc., "The PHANToM" literature from SensAble Devices Inc. 225 Court St., Vanceburg, KY 41179.

[3] Gutierrez T., Barbero J.I., Aizpitarte M., Carrillo A. R., and Eguidazu A., "Assembly Simulation Through Haptic Virtual Prototypes", Proceedings of the Third PHANTOM Users Group Workshop, Cambridge, MA, 1998

[4] Matsumoto S., Fukuda I., Morino H., Hikichi K., Sezaki K., Yasua Y., "The Influences of Network Issues on Haptic Collaboration in Shared Virtual Environments", Proceedings of the Fifth PHANTOM Users Group Workshop, Cambridge, MA, 2000

[5] Hespanha J., McLaughlin M., Sukhatme G., Akbarian M., Garg R., Zhu W., "Haptic Collaboration over the Internet", Proceedings of the Fifth PHANTOM Users Group Workshop, Cambridge, MA, 2000

[6] Watsen K., Zyda M., "Bamboo- a portable system for dynamically extensible, real-time, networked, virtual environments", Virtual Reality Annual International Symposium. Proceedings, Atlanta, GA, 1998